2100030336

M. Gayathri Ankitha

.NET PROGRAMMING

LAB 3

IN-LAB

1. Develop **Rectangle** and **ArrayRectangles** with a predefined functionality.

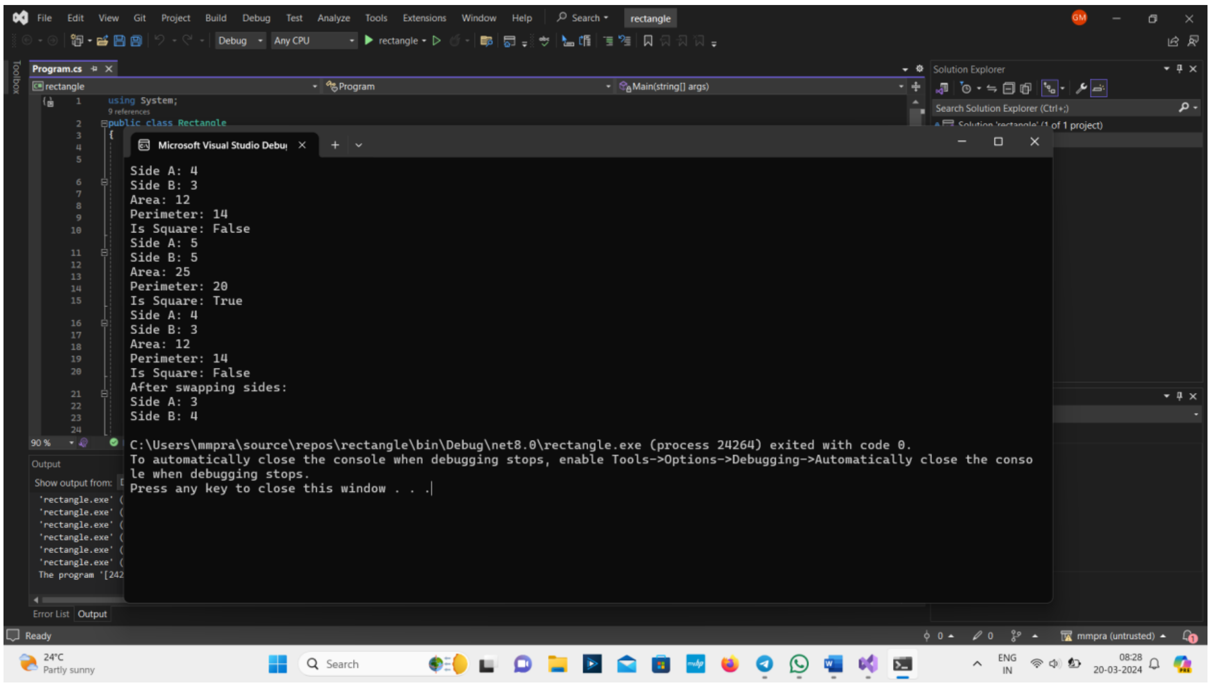
Low level Task:

TASK 1: To develop **Rectangle** class with following content:

- 2 closed real fields **sideA** and **sideB** (sides A and B of the rectangle)
- Constructor with two real parameters **a** and **b** (parameters specify rectangle sides)
- Constructor with a real parameter **a** (parameter specify side A of a rectangle, side B is always equal to 5)
- Constructor without parameters (side A of a rectangle equals to 4, side B - 3)
- Method **GetSideA**, returning value of the side A
- Method **GetSideB**, returning value of the side B
- Method **Area**, calculating and returning the area value
- Method **Perimeter**, calculating and returning the perimeter value
- Method **IsSquare**, checking whether current rectangle is shape square or not. Returns true if the shape is square and false in another case.
- Method **ReplaceSides**, swapping rectangle sides

```
using System;
public class Rectangle
{
    private double sideA;
    private double sideB;
    public Rectangle(double a, double b)
    {
        sideA = a;
        sideB = b;
    }
    public Rectangle(double a)
    {
        sideA = a;
        sideB = 5;
    }
```

```
        public Rectangle()
        {
            sideA = 4;
            sideB = 3;
        }
        public double GetSideA()
        {
            return sideA;
        }
        public double GetSideB()
        {
            return sideB;
        }
        public double Area()
        {
            return sideA * sideB;
        }
        public double Perimeter()
        {
            return 2 * (sideA + sideB);
        }
        public bool IsSquare()
        {
            return sideA == sideB;
        }
        public void ReplaceSides()
        {
            double temp = sideA;
            sideA = sideB;
            sideB = temp;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            Rectangle rectangle1 = new Rectangle(4, 3);
            Console.WriteLine("Side A: " + rectangle1.GetSideA());
            Console.WriteLine("Side B: " + rectangle1.GetSideB());
            Console.WriteLine("Area: " + rectangle1.Area());
            Console.WriteLine("Perimeter: " + rectangle1.Perimeter());
            Console.WriteLine("Is Square: " + rectangle1.IsSquare());
            Rectangle rectangle2 = new Rectangle(5);
            Console.WriteLine("Side A: " + rectangle2.GetSideA());
            Console.WriteLine("Side B: " + rectangle2.GetSideB());
            Console.WriteLine("Area: " + rectangle2.Area());
            Console.WriteLine("Perimeter: " + rectangle2.Perimeter());
            Console.WriteLine("Is Square: " + rectangle2.IsSquare());
            Rectangle rectangle3 = new Rectangle();
            Console.WriteLine("Side A: " + rectangle3.GetSideA());
            Console.WriteLine("Side B: " + rectangle3.GetSideB());
            Console.WriteLine("Area: " + rectangle3.Area());
            Console.WriteLine("Perimeter: " + rectangle3.Perimeter());
            Console.WriteLine("Is Square: " + rectangle3.IsSquare());
            rectangle3.ReplaceSides();
            Console.WriteLine("After swapping sides:");
            Console.WriteLine("Side A: " + rectangle3.GetSideA());
            Console.WriteLine("Side B: " + rectangle3.GetSideB());
        }
    }
```



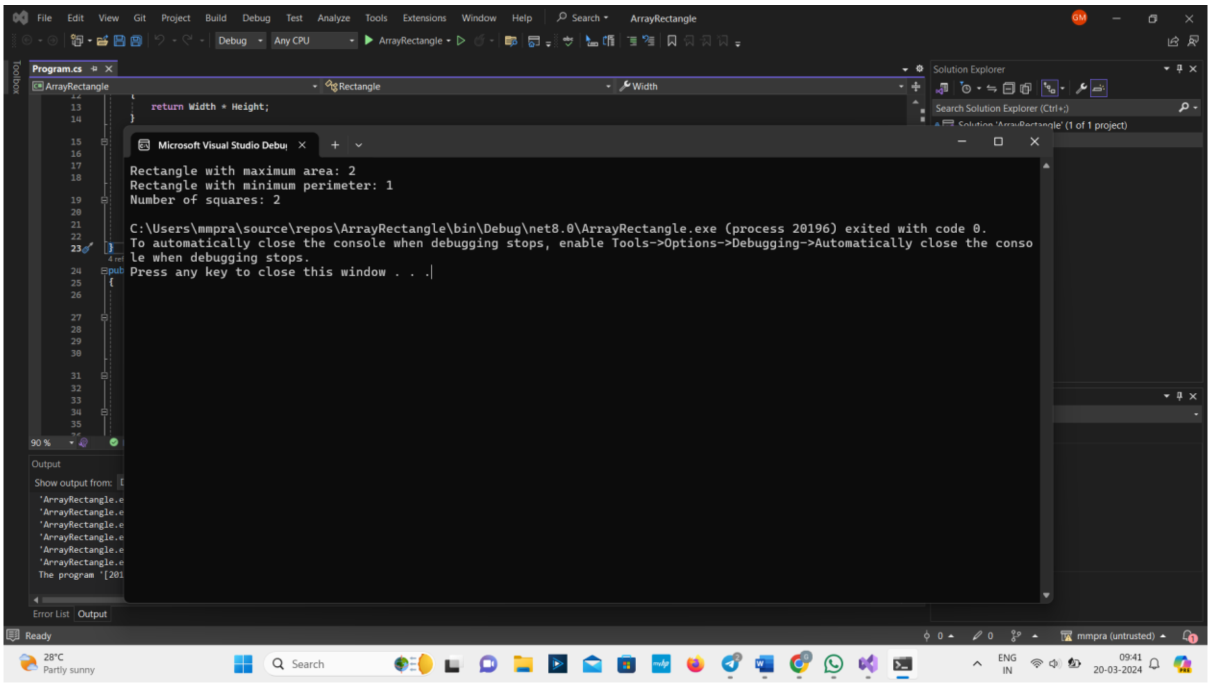
```
Side A: 4
Side B: 3
Area: 12
Perimeter: 14
Is Square: False
Side A: 5
Side B: 5
Area: 25
Perimeter: 20
Is Square: True
Side A: 4
Side B: 3
Area: 12
Perimeter: 14
Is Square: False
After swapping sides:
Side A: 3
Side B: 4

C:\Users\mmpra\source\repos\rectangle\bin\Debug\net8.0\rectangle.exe (process 24264) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Advanced level Task:

TASK 2: Develop class **ArrayRectangles**, in which declare:

- Private field **rectangle_array** - array of rectangles
- Constructor creating an empty array of rectangles with length n
- Constructor that receives an arbitrary amount of objects of type **Rectangle** or an array of objects of type **Rectangle**.
- Method **AddRectangle** that adds a rectangle of type Rectangle to the array on the nearest free place and returning true, or returning false, if there is no free space in the array
- Method **NumberMaxArea**, that returns order number (index) of the rectangle with the maximum area value (numeration starts from zero)
- Method **NumberMinPerimeter**, that returns order number(index) of the rectangle with the minimum area value (numeration starts from zero)
- Method **NumberSquare**, that returns the number of squares in the array of rectangles

```
using System;
public class Rectangle
{
    public double Width { get; }
    public double Height { get; }
    public Rectangle(double width, double height)
    {
        Width = width;
        Height = height;
    }
    public double Area()
    {
        return Width * Height;
    }
    public double Perimeter()
    {
        return 2 * (Width + Height);
    }
    public bool IsSquare()
    {
        return Width == Height;
    }
}
public class ArrayRectangles
{
    private Rectangle[] rectangleArray;
    public ArrayRectangles(int n)
    {
        rectangleArray = new Rectangle[n];
    }
    public ArrayRectangles(params Rectangle[] rectangles)
    {
        rectangleArray = new Rectangle[rectangles.Length];
        for (int i = 0; i < rectangles.Length; i++)
        {
            rectangleArray[i] = rectangles[i];
        }
    }
    public bool AddRectangle(Rectangle rectangle)
    {
        for (int i = 0; i < rectangleArray.Length; i++)
        {
            if (rectangleArray[i] == null)
            {
                rectangleArray[i] = rectangle;
                return true;
            }
        }
        return false;
    }
    public int NumberMaxArea()
    {
        if (rectangleArray.Length == 0)
            return -1;
        double maxArea = rectangleArray[0].Area();
        int maxIndex = 0;
        for (int i = 1; i < rectangleArray.Length; i++)
        {
            if (rectangleArray[i] != null && rectangleArray[i].Area() > maxArea)
            {
                maxArea = rectangleArray[i].Area();
                maxIndex = i;
            }
        }
        return maxIndex;
    }
    public int NumberMinPerimeter()
    {
        if (rectangleArray.Length == 0)
            return -1;
        double minPerimeter = rectangleArray[0].Perimeter();
        int minIndex = 0;
        for (int i = 1; i < rectangleArray.Length; i++)
        {
            if (rectangleArray[i] != null && rectangleArray[i].Perimeter() <
minPerimeter)
            {
                minPerimeter = rectangleArray[i].Perimeter();
                minIndex = i;
            }
        }
        return minIndex;
    }
    public int NumberSquare()
    {
        int count = 0;
        foreach (var rectangle in rectangleArray)
        {
            if (rectangle != null && rectangle.IsSquare())
                count++;
        }
        return count;
    }
}
class Program
{
    static void Main(string[] args)
    {
        Rectangle rectangle1 = new Rectangle(4, 5);
        Rectangle rectangle2 = new Rectangle(3, 3);
        Rectangle rectangle3 = new Rectangle(6, 6);
        ArrayRectangles arrayRectangles = new ArrayRectangles(3);
        arrayRectangles.AddRectangle(rectangle1);
        arrayRectangles.AddRectangle(rectangle2);
        arrayRectangles.AddRectangle(rectangle3);
        Console.WriteLine("Rectangle with maximum area: " +
arrayRectangles.NumberMaxArea());
        Console.WriteLine("Rectangle with minimum perimeter: " +
arrayRectangles.NumberMinPerimeter());
        Console.WriteLine("Number of squares: " +
arrayRectangles.NumberSquare());
    }
}
```



```
        return Width * Height;
    }
```

```
Rectangle with maximum area: 2
Rectangle with minimum perimeter: 1
Number of squares: 2

C:\Users\mmpra\source\repos\ArrayRectangle\bin\Debug\net8.0\ArrayRectangle.exe (process 20196) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

```
'ArrayRectangle.e
'ArrayRectangle.e
'ArrayRectangle.e
'ArrayRectangle.e
'ArrayRectangle.e
'ArrayRectangle.e
The program '[201
```